Introduction to Software Security

The majority of this material is summarized and excerpted from *Security in the Software Lifecycle: Making Software Development Processes – and Software Produced by Them – More Secure* [DHS 06].

Software's Vulnerability to Attack

What makes it so easy for attackers to target software is the virtually guaranteed presence of vulnerabilities, which can be exploited to violate one or more of the software's security properties. According to CERT, most successful attacks result from targeting and exploiting known, non-patched software vulnerabilities and insecure software configurations, many of which are introduced during design and code.

In their Report to the President titled *Cyber Security: A Crisis of Prioritization* [PITAC 05], the President's Information Technology Advisory Committee summed up the problem of non-secure software as follows:

> Software development is not yet a science or a rigorous discipline, and the development process by and large is not controlled to minimize the vulnerabilities that attackers exploit. Today, as with cancer, vulnerable software can be invaded and modified to cause damage to previously healthy software, and infected software can replicate itself and be carried across networks to cause damage in other systems. Like cancer, these damaging processes may be invisible to the lay person even though experts recognize that their threat is growing. And as in cancer, both preventive actions and research are critical, the former to minimize damage today and the latter to establish a foundation of knowledge and capabilities that will assist the cyber security professionals of tomorrow reduce risk and minimize damage for the long term.

The software development process offers opportunities to insert malicious code and to unintentionally design and build software with exploitable weaknesses. Security-enhanced processes and practices—and the skilled people to perform them—are required to build software that can be trusted not to increase risk exposure.

The Challenge of Building Secure Software

Secure software is software that is resistant to intentional attack as well as unintentional failures, defects, and accidents. Software security is the ability of software to resist, tolerate, and recover from events that intentionally threaten its dependability.

Security in the Software Lifecycle [DHS 06] defines software security as follows:

> Protection against intentional subversion or forced failure. Preservation of the three subordinate properties that make up security—availability, integrity, and confidentiality.
>
> Security manifests as the ability of the system to protect itself from external faults that may be accidental or deliberate (attacks). According to Bruce Schneier in *Beyond Fear* [Schneier 06], "Security is about preventing adverse consequences from the intentional and unwarranted actions of others."

The objective of software security is to design, implement, configure, and support software systems in ways that enable them to

1. continue operating correctly in the presence of most attacks by either *resisting* the exploitation of

faults or other weaknesses in the software by the attackers or *tolerating* the errors and failure that result from such exploits

2. isolate, contain, and limit the damage resulting from any failures caused by attack-triggered faults that the software was unable to resist or tolerate and recover as quickly as possible from those failures

Software Assurance

Software security and secure software are often discussed in the context of software assurance. Software assurance is broader than software security, encompassing the additional disciplines of software safety and reliability.

A key objective of software assurance is to provide justifiable confidence that software is free of vulnerabilities. Another is to provide justifiable confidence that software functions in the "intended manner" and the intended manner does not compromise the security and other required properties of the software, its environment, or the information it handles.

A third objective of software assurance is the ability to trust, with justified confidence, that software will remain dependable under all circumstances. These include

- the presence of unintentional faults in the software and its environment
- exposure of the operational software to accidental events that threaten its dependability
- exposure of the software to intentional threats to its dependability, both in development and in operation

According to the Committee on National Security Systems (CNSS) Instruction No. 4009, "National Information Assurance Glossary[1]," software assurance is defined as

> the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner.

Software assurance addresses these attributes:

- Trustworthiness: no exploitable vulnerabilities exist, either maliciously or unintentionally inserted
- Predictable execution: justifiable confidence that software, when executed, functions as intended
- Conformance: planned and systematic set of multidisciplinary activities that ensure that software processes and products conform to requirements and applicable standards and procedures

Software Security vs. Information Security

The distinction between software security and information security lies in the inherent difference between software and information. Information is passive by nature; it cannot perform actions; it can only have actions performed upon it. How the information is created, and whether it is correct or not, has no effect on the ability to secure it. A crucial difference between software and information that makes achieving security such a different proposition for each is that information does not become vulnerable because of how it was created. It becomes vulnerable only because of how it is stored or transmitted. Information systems security measures focus exclusively on counteracting those threats to the system that if successful would compromise the information within the system.

Unlike information, software can exist simultaneously as a passive file stored on a file system and in an

1. http://www.cnss.gov/instructions.html

active, executing state. Because of its multifaceted nature, software can be vulnerable because of the way in which it was created (developed).

In summary, information needs to be secure because of what it is and how it is acted upon by other entities. Software needs to be secure because of what it does, including how it acts upon other entities. The main objective of information security and the systems that store and transmit information is to protect information from unauthorized disclosure, modification, or deletion. The main objective of software security is to produce software that will not be vulnerable to unauthorized modification or denial of service during its execution.

The Qualities of Secure Software

The vulnerabilities in executing software originate in the process used to create that software: the decisions made by its developers, the flaws they inadvertently or intentionally include in its specification and design, and the faults and other defects they inadvertently or intentionally include in its implemented code.

In addition to trustworthiness, predictable execution, and conformance, secure software must be attack-resistant or attack-tolerant, and at the whole system level it must be attack-resilient.

To achieve attack-resistance or attack-tolerance, both software components and whole software systems should be able to recognize attack patterns in the input data or signals they receive from external entities (humans or processes). They should be able to either resist attack-patterned input or tolerate the failures that result from a successful attack or intentional external fault.

To achieve attack-resilience (often referred to as survivability), software systems must be able to recover from any failures that result from successful attacks on the software by resuming operation at or above some predefined minimum acceptable level of service in the short term. The system must eventually recover full service at the specified level of performance.

Attributes of security as a property of software include availability, integrity, confidentiality, accountability, and non-repudiation (refer to [DHS 06] for definitions of these).

Deploying Software Security Practices

The main goals of deploying software security practices include the following:

- Exploitable faults and other weaknesses are avoided by well-intentioned developers.
- The likelihood is greatly reduced or eliminated that malicious developers can intentionally implant exploitable faults and weaknesses or malicious logic into the software.
- The software is attack-resistant, attack-tolerant, and attack-resilient.

The best practices, knowledge, and tools articles on the Build Security In Web site support organizations in making progress toward achieving these goals. Those responsible for ensuring that software and systems meet their security requirements throughout the development life cycle should review, select, and tailor BSI guidance as part of normal project management activities. Additional Resources[2] on BSI and the references below provide additional, experience-based practices and lessons learned that development organizations need to consider.

References

2. daisy:439 (Additional Resources)

[DHS SWA] DHS Cyber Security. Software Assurance[3]. Department of Homeland Security Infosheet.

[DHS 06] Department of Homeland Security. *Security in the Software Lifecycle: Making Software Development Processes – and Software Produced by Them – More Secure*[4], draft version 1.2. DHS, August 2006.

[Howard 06] Howard, Michael & Lipner, Steve. *The Security Development Lifecycle; SDL: A Process for Developing Demonstrably More Secure Software*. Redmond, WA: Microsoft Press, 2006.

[McGraw 06] McGraw, Gary. *Software Security: Building Security In*. Boston, MA: Addison-Wesley, 2006.

[PITAC 05] President's Information Technology Advisory Committee. *Cyber Security: A Crisis of Prioritization: Report to the President*[5]. National Coordination Office for Information Technology Research and Development, February 2005.

[Schneier 06] Schneier, Bruce. *Beyond Fear*. Heidelberg, Germany: Springer-Verlag, 2006.

3. http://www.us-cert.gov/reading_room/infosheet_SoftwareAssurance.pdf
4. daisy:87 (Security in the Software Lifecycle)
5. http://www.nitrd.gov/pitac/reports/